

# Google Commerce Search

## Deployment Guide

September 2011



© 2011 Google

# Deployment Guide

This guide is meant to answer many common questions about Google Commerce Search (GCS).

## About this document

The recommendations and information in this document were gathered through our work with a variety of clients and environments in the field. We thank our customers and partners for sharing their experiences and insights.

<b>What's covered</b>	This guide provides some general guidance and best practices around Google Commerce Search.
<b>Primary audience</b>	Developers or technical architects involved with implementing GCS for a particular web site.
<b>Deployment phases</b>	Initial and ongoing deployment of GCS.
<b>Other resources</b>	<ul style="list-style-type: none"><li>• <a href="#">GCS product documentation</a> provides complete information about API usage and general administration.</li><li>• <a href="#">Search API for Shopping documentation</a> provides information about using this API.</li><li>• <a href="#">Content API for Shopping documentation</a> provides information about using this API.</li><li>• <a href="#">Enterprise Support Portal</a> provides access to Google support.</li></ul>

## **Contents**

[About this document](#)

[Chapter 1 Understanding Your Deployment](#)

[Administration](#)

[APIs Console](#)

[Deployment](#)

[Chapter 2 Merchant Center, Feeds and Content API](#)

[Structuring your content](#)

[Chapter 3 The GCS Control Panel](#)

[Admin accounts](#)

[Automating admin tasks](#)

[Chapter 4 The Search API for Shopping](#)

[Request format](#)

[Response format](#)

[Chapter 5 GCS Stats and Analytics](#)

[GCS Statistics page](#)

[API Console Traffic Reports](#)

[Google Analytics integration](#)

[Appendix A Limits](#)

[Items in general](#)

[Feed Size](#)

[Content API Limit](#)

[Search Limit](#)

[Appendix B GCS Launch Checklist](#)

## Chapter 1 Understanding Your Deployment

Google Commerce Search (GCS) is a hosted search product--a web service that can provide search results on your web site. You have complete control over the look and feel of the user interface, as well as the ability to pick and choose which features to enable. Because GCS is a web service, and uses open, standard protocols (HTTP, XML, JSON), the end-user experience can easily be customized to provide anything from basic search results to complex dynamic user interfaces that allow users to really interact with many different search components.

Integrating Google Commerce Search with an existing site involves making some changes to use GCS to handle all of your site-search requests, and possibly navigation or browsing too. If GCS is replacing an existing search provider on your site, the basic integration process may look something like this:

1. Modifying the existing product feed to add more custom item attributes (Merchant Center).
2. Modifying the existing search box (html form) to point to new GCS script (that is, /search.php).
3. Installing the new GCS search script/app.  
Example code for php, java and .Net can be found here:  
<http://code.google.com/p/gcs-admin-toolkit/>
4. Modifying the output of new search results to match current site look and feel, typically through CSS/stylesheet changes.
5. Applying business/merchandising rules by using the control panel.
6. Adding Query Suggestions/Search-as-you-type to search box, if desired.

### Administration

There are two tools used for the administration of GCS:

- [Merchant Center](#)
- [Commerce Search Control Panel](#)

#### Merchant Center

Merchant Center (<http://www.google.com/merchants>) is used to manage the feed process. Before you can use GCS, you must feed your content to Google. Merchant Center provides the tools to manage your feeds and view the raw data for your products. Merchant Center is discussed in more detail in [Chapter 2](#).

#### Commerce Search Control Panel

The Commerce Search Control Panel (<http://www.google.com/cse/commercesearch/manage>) is used to control the search experience--basically anything that has to do with GCS search results on your site. The control panel does not directly control the look and feel of your search results, but it controls the merchandising features that affect those results. The Control Panel is discussed in detail in [Chapter 3](#).

You can associate any email address with a Google account (google.com/account), and then use that account to log in to Merchant Center or the GCS control panel. Merchant Center currently has no way to delegate or add other accounts for login, so if multiple people do need to access Merchant Center you will need to share those login credentials. The GCS control

panel does allow additional accounts to be added for access, so sharing credentials shouldn't be needed.

## APIs Console

The APIs Console (<https://code.google.com/apis/console/>) is a third tool used in conjunction with GCS.

To use GCS, you must have an API key--many Google services rely on developer keys to control access. A key for the "Search API for Shopping" is required when creating your GCS instance. By default, your key is limited to 2,500 queries per day--this needs to be raised by Google Support before putting your search engine into production. Typically, your API key is provided during the initial account provisioning. You can verify your daily quota in the Quotas section of the API console.

**Best Practice:** Use two different API keys:

- One key for any client side code, such as Search As You Type (SAYT), Query Autocompletions, and so on
- One key for your server-side code

This practice enables you to use Traffic Filters (in Quotas section) efficiently to control which domains or servers can utilize your API keys.

**Best Practice:** For server-side code, increase your per-user limit. Because all requests will be coming from a limited number of IP addresses (your servers), these appear as single "users." Your per-user rate should be enough to handle peak queries per second (QPS).

## Deployment

The actual deployment of GCS involves installing some code on your web site. The bulk of this code is typically in the form of a server-side application or script. Although it is possible to implement GCS with browser-based, client-side code only (Javascript), that method is strongly discouraged.

**Best Practice:** Use server-side code (php, java, .Net, and so on) for your search pages. While the Shopping API can return JSON and be implemented in JavaScript, this tends to be very hard to maintain.

While Javascript may have an advantage of speed (results served directly from Google API to end users), it is typically harder to maintain and debug. Current JavaScript frameworks, such as JQuery and GWT, relieve many of the cross-browser headaches, but some functionality is lost or harder to implement. For example, the back button often no longer functions, so users can't easily go back to previous searches/pages, and bookmarkable URLs are harder to achieve.

That said, the API does support JSON and can be used to deploy a fully or partially client-side search engine implementation if you are comfortable doing so. However, only server-side deployments are fully supported by Google's Enterprise Support team.

The server-side code performs the following steps:

1. Handles the search form request when a user submits a search query on your site

2. Formats a request for the Google Shopping API based on the keywords and any other parameters specified in the search request
3. Sends the Request to Google
4. Receives the Shopping API response from Google
5. Parses the Response
6. Applies any HTML or CSS styles to the results

Results are displayed in the user's browser.

While it might sound like a lot of work, this all happens in under a second, in most cases. The request/response format is typical to any Web Services-based application, and is done by way of HTTP. It's RESTful, so no state needs to be maintained between requests.

Example implementation code for php, Java (jsp) and .NET is available as part of the GCS Admin Toolkit project at <http://code.google.com/p/gcs-admin-toolkit/>

Implementation of the server-side code using the Shopping API is discussed in [Chapter 4](#).

## Chapter 2 Merchant Center, Feeds and Content API

Unlike Google Site Search, which uses the Google crawler (googlebot) to index your site, GCS builds its index from content that is directly fed or “pushed” to Google. In most cases, you might already be feeding some of your product data to Google through Merchant Center, for Google Product Search (formerly called “Froogle”). This product feed can also be shared with Google Commerce Search. Likewise, you can specify that a feed is only for GCS, and the items will not show up in public product search.

The Merchant Center feed defines a number of required and recommended fields, as described in the “Product Feeds Specification” at <http://www.google.com/support/merchants/bin/answer.py?answer=188494>

You can also find examples of the various feed formats at this link.

GCS has the same required attribute fields for each item: id, title, link, price, description, condition. Any of the recommended fields can also be used in your GCS implementation. But as a GCS customer, you are also able to submit custom-defined attributes.

### Structuring your content

Before we look at the technical details involved with feeding content to Google, it is important first to map your current product schema into something GCS can use. Typically, this exercise involves determining which fields in your current database are important for search functionality, and then mapping them to fields in the feed sent to Google.

Here are some questions that will help you determine what data you want to send to Google:

- What data do you need to display in search results?  
For example, thumbnail image, title, and price
- Which fields do you need to use for refinements?  
For example, color, size, and brand
- Which fields do you need to use for sorting?  
For example, price, date added, and rating
- Which fields might you use for ranking?  
For example, best seller, quantity in stock, and featured item
- Do you need to do any range-based searches?  
For example, dates, prices, sizes

Essentially, for any features you wish to implement, the supporting data needs to be submitted in your feed.

A typical GCS data feed might contain the following attributes for most items:

Attribute	Type	Description
id	required	Think of id as your primary key. This has to be a unique identifier. It doesn't have to be a number - it can be a SKU or product id.
link	required	The URL of the item.
title	required	Product Title, up to 70 characters.
description	required	Product Description, up to 10K characters Almost exclusively used for search, not display, so formatting is typically not that important.
price	required	If you have tiered pricing, use List Price here and custom attributes to define other prices or a price range.
condition	required	New or Used.
image_link	required (for PS)	Image URL for the Full Size Image.
quantity	primary	Quantity in Stock. Useful for ranking rules or sorting.
brand	primary	Brand or Manufacturer of product.
color	primary (multi-value)	Colors that item is available in (Red/Black/Green, etc.).
size	primary (multi-value)	Sizes that item is available in (Small/Med/Large, etc.).
thumbnail	custom	Link to thumbnail Image for search results.
featured	custom	Possibly used for ranking rule, or to display as a different style.

### Product variants

Product variants are options or attributes for a product, but typically they reference the same top-level productid or sku. For example you might have an "Android Polo," but it is available in black or white, and in small, medium and large.

For Google Product Search, you might submit a separate URL for each variant of a product, but with GCS you should just submit the variants as additional attributes to the same product. This way, when someone searches for "android polo" they don't get 6 items back, they just get the one. Likewise, even if they search for "small android polo" they would still get the same result, because "small" is a searchable attribute of the item.

## Merchant Center overview

The Merchant Center console enables you to specify and upload feeds to Google Product Search and Google Commerce Search. It also provides some diagnostics and reporting for your feeds and product data. Typically, you won't need to log in to Merchant Center much once your feeds are submitted and working properly.

If you're not currently submitting feeds, you will probably want to click the **New Data Feed** button in the **Data Feeds** tab, otherwise click on the **Edit** link next to your existing feed file name to change any of its parameters. This will be a product feed **Type**, of "googlebase" format. The **Data feed filename** you specify should match the name of the file you plan to upload. For large files, you should plan on compressing them, so you might enter a filename, such as `googlefeed.xml.gz`.

The screenshot shows the 'New data feed' form in the Merchant Center console. On the left is a navigation menu with links for Dashboard, Data feeds (selected), Products, Data quality, Performance report, and Settings. The main content area is titled 'Data feeds > New data feed'. It contains a descriptive paragraph, a 'Type' dropdown menu set to 'Products', a 'Target country' dropdown menu set to 'United States', and a 'Format' dropdown menu set to 'googlebase'. Below these is a note to read guidelines, a 'Data feed filename' text input field containing 'googlefeed.xml.gz', and another note about special characters. A link for 'Set up advanced feed usage settings >>' is present. Underneath are four checked checkboxes for 'Commerce Search', 'Product Ads', 'Product Search', and 'Search API for Shopping'. At the bottom are 'Save changes' and 'Cancel' buttons.

Click the **Set up advanced feed usage settings** link and you can see checkboxes for:

- **Commerce Search**
- **Product Ads**
- **Product Search**

If you want to enable this feed for GCS, make sure **Commerce Search** is checked as a Target Property. If you do not see the **Commerce Search** option, contact Google Support with your Account ID and they will enable GCS for your Merchant Center Account.

If you are using the [Content API for Shopping](#) to update your items, these target properties correspond to target "Destinations" (or use cases), as described in the following table. One benefit of using the Content API versus feeds, is that you have the ability to specify the target at

an item level instead of the entire feed.

Destination Name	API Name	Description
Google Product Search	ProductSearch	Your products will appear on Google Product Search (Set by default).
Product Ads	ProductAds	You can use your product data to highlight your products in your Google.com search ads. <a href="#">Learn more.</a>
Commerce Search	CommerceSearch	You can use your product data for our new service which utilizes Google's search technology to power your own retail site. <a href="#">Learn more.</a>

You can create a schedule to have Google fetch the feeds from your web server at regular intervals. For testing purposes, you can also do manual uploads for feeds under 20MB directly in the admin console. Pushing feeds to Google by way of FTP is also a good option. Under the **Settings > FTP** tab you can set the username and password to use. To automate FTP, you can use a `.netrc` file and a `cron` script on Unix servers, or there are a number of other ways that FTP uploads can be automated. For fetch and FTP, your files must be under 1GB (500MB if compressed).

### Feed formats

Feeds can be submitted in text (tab-separated) or XML formats. XML is the preferred choice, but if you are currently submitting a text feed and only need to add a few custom attributes for GCS, then it probably isn't necessary to switch to XML. Text format is fine for small sites, with many similar products, but it becomes harder to manage when you have a wide variety of products. Text formatting require the same number of fields be submitted for each product, so if you have attributes that only apply to some products those fields would still need to be submitted as empty fields for other products. With XML, you are only submitting exactly the attributes you need for each product.

Custom Attributes are defined for text files at:

<http://base.google.com/support/bin/answer.py?hl=en&answer=59463>

and for XML format at:

<http://base.google.com/support/bin/answer.py?answer=59558&hl=en>

**Best Practice:** Use the same feed for Products Search and GCS. Also, the URL you submit to Merchant Center for each item should be the same URL that is found through your site navigation.

### Test data feeds

If you plan on using the same data feed for product search and GCS, there is a common concern about testing the feed before submitting to production. Because each feed does a "replace" and not an "addition," if there are errors with items in the feed they will get removed from the index. To aid in the development of your feed, and avoid errors in production, you can create a test data feed to validate the format. These feeds are processed the same way as

regular feeds, but the items are not inserted into the index. If there are errors, you will see them on the feed status page.

The test feeds are a good way to work out any issues with your formatting when adding any new fields to the feed. Once the feed submits without error you should be able to submit a production feed without problem.

### Product status and data quality

Once your feed has been processed, new items are added to the index and updates to existing items occur. This process may take several hours for a large feed. Existing items remain searchable, so there is no downtime while the feed is processing. Each item is processed individually, so some changes may be seen before others--the feed is not processed completely and then “cut-over.”

To verify the status of any item, you can check the **Products** tab in Merchant Center. When an item is done processing and in a searchable state, you should see a green check mark in the **Commerce Search** column:

<input type="checkbox"/>	Title	Country	Language	Product Search <sup>?</sup>	Product Ads <sup>?</sup>	Commerce Search <sup>?</sup>
<input type="checkbox"/>	<a href="#">Central Y. M. C. A., 1421 Arch...</a>	United States	English	--	--	✓
<input type="checkbox"/>	<a href="#">Golden Gate Bridge Across Gold...</a>	United States	English	--	--	✓
<input type="checkbox"/>	<a href="#">Big One Got Away Fishing Exagg...</a>	United States	English	--	--	✓
<input type="checkbox"/>	<a href="#">Deer Leap Rutland, VT</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Champlain Street Quebec</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">The Henry Ford Hospital Detroi...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Crystal Lake Eastford, CT Roya...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Baby New Year H B Griggs HBG R...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">A Roadway In The Trossachs Sco...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">General View Arrowhead Hot Spr...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Sunriver Lodge Bend, OR</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Town Motel Titusville, FL</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">First Lesson, the Seminole Ind...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Greetings From Bristol Bristol...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">View From The Manchester Bridg...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Collings Summer Home, Turner F...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Greetings From New York City N...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Spaulding High School Rocheste...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Pryor's Drive-In Restaurant Lo...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">United States Post Office Samp...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Horses on Rotten Row in Hyde P...</a>	United States	English	✓	✓	✓
<input type="checkbox"/>	<a href="#">Garfield Park Chicago, IL</a>	United States	English	✓	✓	✓

You can search on the product id to find details on specific products. Clicking on the individual product link, you can view all of the attributes that were submitted:

Dashboard  
Data feeds  
**Products**  
Data quality  
Performance report  
⊕ Settings

Products >  
YouTube Organic cotton T-shirt - White

Item URL  
http://www.googlestore.com/Eco/YouTube+Organic+cotton+T-shirt+-+White...

Pictures



Details

<b>Id:</b>	1013043
<b>Price:</b>	\$11.85
<b>Quantity:</b>	1
<b>Thumb link:</b>	http://www.googlestore.com/content/images/thumb/10-13043b.jpg
<b>Theimage:</b>	http://www.googlestore.com/content/images/standard/10-13043a.jpg
<b>Condition:</b>	NEW
<b>Gsgender:</b>	Unisex
<b>Shipping weight:</b>	0.75 lbs
<b>Bucket:</b>	2
<b>Gssize:</b>	S
<b>Gssize:</b>	M
<b>Gssize:</b>	L
<b>Gssize:</b>	XL
<b>Gssize:</b>	2XL
<b>Gssize:</b>	3XL
<b>Color:</b>	WHITE
<b>Eco:</b>	Organic
<b>Gscategory:</b>	Wearables
<b>Gscategory:</b>	Eco
<b>Gscategory:</b>	You Tube
<b>Isnew:</b>	no
<b>Topseller:</b>	no
<b>Specials:</b>	no

Description

Embellish your fashion statement with YouTube's full color logo on a 100% pre-shrunk certified organic cotton t-shirt. Features include shoulder to shoulder taping. Comes in white with a full color YouTube logo screened on front of shirt.

The Data Quality tab appears.

## Content API for Shopping

The [Content API for Shopping](#) lets you programmatically manage your catalog instead of relying on a feed file. It allows for much greater control of your products where you can selectively set usecases, inventory, expire products, delete and update items. It offers much more flexibility and provides item-by-item feedback if a particular operation (insert/update/delete) succeeded or failed.

As with other authenticated Google APIs, the Content API is a RESTful web service that requires a security token for each operation. The security token is derived from a [Google ClientLogin](#), which may be reused for multiple operations. An application wishing to insert an item will first have to get a client login token as the user who owns the Merchant Center account, then take the token and perform the insert operation. The Content API response will show if the particular item passed validation and if it's staged for insert.

## Content API operations

The various operations the Content API supports are:

- insert
- delete
- update
- batch
- query (just to recall or iterate the catalog)

All operations are over HTTPS using various HTTP verbs (GET, POST, PUT, DELETE) and

XML is transmitted as the payload. For example operations, see the Content API docs or the [GCS Docs for Feeds](#).

Unlike feed files, you must explicitly issue delete commands for a given item to remove it from the inventory. With feed files, if an item is not present in subsequent feed files, it is considered void and is deleted from GCS tables. With the ContentAPI, an item remains in the GCS tables for a default 30 days unless it is explicitly deleted.

You may want to start off testing the Content API by using the [Interactive Demo](#) or the sample RAW protocol examples on the [GCS Admin Toolkit](#). Take note that you must be logged in as the owner of the Merchant Center account to use the Interactive Demo and the username/password provided to the sample Java application must also be the owner.

The normal processing time for new items inserted by way of the ContentAPI is slightly faster than feed files but may take up to 12 hours. Updating existing items may also take just as long. However, there is an “Express indexing pipeline” which allows for rapid price or quantity updates in 15 minutes. If you have an existing item that is serving, you may run an insert or update for the entire item definition with just the price or quantity as the difference. GCS will then process this item under the express pipeline and index/serve with the new price within 15 mins. Take note that to use the express pipeline (or any update/insert operation for that matter), you must transmit an entire, complete item definition.

To manage large inventories, Google recommends that you use the batch protocol for multiple operations. You can combine inserts/updates/deletes in one file and the response from the ContentAPI will describe the success/failure for each operation. The maximum file size for the batch protocol is 1MB.

#### **Miscellaneous notes about the API and its usage**

- There is no need to acquire clientlogin tokens for each operation; one may be reused for up to 28 days
- Setting and using the `expiration_date` attribute for an item will not delist an item at exactly at that time. It may take up to a day or two to delist.
- The ContentAPI currently only supports product management; the category feeds is currently not supported.

## Chapter 3 The GCS Control Panel

This chapter doesn't comprehensively document all the features in the control panel, rather it highlights some best practices and specific use cases relative to the control panel. For complete information, see:

- GCS Configuration pages: <http://code.google.com/apis/commercesearch/docs/gcsconfiguration.html>
- Context-sensitive help within the control panel

### Admin accounts

There can be only one owner of a GCS instance associated with a single Google account. But there can be many administrators. The owner can delegate access to other users by using the **Admin Accounts** tab. Take note that before you can add a user as an admin, their email must be associated with a Google account at <https://www.google.com/accounts/>.

If you need to change the owner of the GCS instance, you need to file a ticket with Google Enterprise Support at <http://www.google.com/enterprise/portal/>.

### Audit logs

While there is no way to rollback changes to GCS settings, you can get a list of changes that have been made by GCS administrators in the **Audit Log** tab. There are currently no notifications sent when changes are made, but this functionality could be implemented using the [Admin API](#) to poll for changes on a regular basis.

### Automating admin tasks

While most users will administer GCS by using the user interface of the control panel, in some cases it may be easier either to automate tasks using the admin API, or to use the import/export functionality to import machine- or manually-generated XML files.

### XML import/export

For one-off bulk uploads, or tasks that are fairly infrequent, the XML-import feature is very easy to use. Most of the features in the control panel support uploading and downloading of their configurations. Currently all of these features support exporting and importing their settings individually through the control panel:

- Synonyms
- Autocompletions
- Promotions
- Ranking Rules
- Facet Rules
- Redirect Rules
- Attributes (Display Names)

Alternatively you can also download the entire configuration of your search engine through the **Advanced > Configuration** tab.

If you have hundreds of synonyms, or need to manage many different promotions or rules, you can create an XML file and upload them all at once, rather than manually entering the data in

the control panel. To see examples of the XML:

1. Create some rules manually.
2. Download the resulting XML.
3. Open the XML into your favorite editor, or have a script generate more XML in the same format.

## Admin API

The GCS control panel shares the same API as CSE (Custom Search Engine), and most of the functionality within the control panel can be accessed programmatically using this API. The CSE API is documented at <http://code.google.com/intl/en/apis/customsearch/docs/api.html>.

Some things to note:

- All administrators/collaborators to the search engine have the ability to use the API.
- Not all of the CSE elements translate:
  - Example for autocomplete: the `authtoken` needs to have `service=cprose`
  - <http://code.google.com/p/gcs-admin-toolkit/source/browse/trunk/src/apiinsert/ClientLogin.java>

```
~/Desktop/auto$ more data.txt
<Batch>
<Add>
<Autocompletions total="1">
<Autocompletion term="someword" type="1" language="" match="1"/>
</Autocompletions>
</Add>
</Batch>
```

```
~/Desktop/auto$ curl -v -trace -H "Authorization: GoogleLogin auth=DQAAAKoABCAJvYrL4I-
uDVX0O8cTgisW4fuPHN0wzjFLb9Gn_JHsSuMGwFkcCZRS9dpilysAEnDrmfoiajxKHGN0fvLOKXnwePSfXTqejp3I-
S0CG6tWDGkLVw-
NGoioe1nkT4uTdgTwmJgXekI6yO7wBHUQQmwLMBB0RpV0d7eXNZMKuc3LTakEMmat193xOieboyhclnGD8y7IxGeP
TW_ArE3H5ojuS6nV6LUp8x98LJfenrFCoQ" -X POST -d @data.txt -H "Content-type: text/xml" "http://www.google.com/
cse/api/014034025922634248784/autocomplete/udhuvsynrfw?num=20&start=0&actype=1&acmatch=1"
* About to connect() to www.google.com port 80 (#0)
* Trying 2001:4860:b006::6a... connected
* Connected to www.google.com (2001:4860:b006::6a) port 80 (#0)
> POST /cse/api/014034025922634248784/autocomplete/udhuvsynrfw?num=20&start=0&actype=1&acmatch=1 HTTP/
1.1
> User-Agent: curl/7.19.7 (x86_64-pc-linux-gnu) libcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15
> Host: www.google.com
> Accept: */*
> Authorization: GoogleLogin auth=DQABCKoAAAAJvYrL4I-
uDVX0O8cTgisW4fuPHN0wzjFLb9Gn_JHsSuMGwFkcCZRS9dpilysAEnDrmfoiajxKHGN0fvLOKXnwePSfXTqejp3I-
S0CG6tWDGkLVw-
NGoioe1nkT4uTdgTwmJgXekI6yO7wBHUQQmwLMBB0RpV0d7eXNZMKuc3LTakEMmat193xOieboyhclnGD8y7IxGePT
W_ArE3H5ojuS6nV6LUp8x98LJfenrFCoQ
> Content-type: text/xml
> Content-Length: 139
>
< HTTP/1.1 200 OK
< Date: Tue, 10 May 2011 00:40:36 GMT
< Expires: Fri, 04 Aug 1978 12:00:00 GMT
< Cache-Control: private, no-cache, no-cache="Set-Cookie", proxy-revalidate
< Content-Type: text/xml; charset=UTF-8
< Pragma: no-cache
```

< Set-Cookie: PREF=ID=4debb263401e6f34:TM=1304988036:LM=1304988036:S=XPruSwy4M\_04MWrl; expires=Thu, 09-May-2013 00:40:36 GMT; path=/; domain=.google.com  
< X-Content-Type-Options: nosniff  
< Server: pfe  
< X-XSS-Protection: 1; mode=block  
< X-Google-Backends: /bns/ia/borg/ia/bns/prose-team/prose.frontend/1,pvd39:80  
< X-Google-GFE-Request-Trace: pvd39:80,/bns/ia/borg/ia/bns/prose-team/prose.frontend/1,pvd39:80  
< X-Google-GFE-Service-Trace: custom-search-controlpanel  
< X-Google-Service: custom-search-controlpanel  
< X-Google-GFE-Response-Body-Transformations: gunzipped,chunked  
< Transfer-Encoding: chunked  
<  
<?xml version="1.0"?>  
<Batch>  
<Add>  
<Autocompletions total="1">  
 <Autocompletion term="someword" type="1" language="" match="1"/>  
</Autocompletions>  
</Add>  
</Batch>

## Chapter 4 The Search API for Shopping

The Search API for Shopping (Shopping API) is what allows you to search the products fed into Merchant Center. The Shopping API is typically implemented in server-side code (Java, php, .Net, and so on) which takes a user's search query, makes an API call to Google, and gets the search results back--presenting them to the user.

This chapter covers some of the common use cases for the Shopping API and discusses overall implementation architecture, but it does not give specific coding examples. The Shopping API provides a rich set of features, and there are a number of ways that a search application can be implemented. For example code, see the GCS Admin Toolkit at

<http://code.google.com/p/gcs-admin-toolkit/>

**Best Practice:** Use a GET request for your search form--this allows easy tracking of query parameters via Google Analytics. It also allows users to easily bookmark search results pages.

### Request format

The Shopping API has a wealth of features that enable you to build robust search interfaces. It is a RESTful API, so the results of an API call are controlled by the parameters in the URL of the request. Some of the things these parameters control are:

- Number of results returned, pagination options
- Sorting order of results
- Which attributes, facets or fields are returned
- Refinement & Filtering options
- Format of results (XML, JSON)

This chapter covers some of these parameters, but for detailed technical documentation for the Shopping API, go here:

<http://code.google.com/apis/shopping/search/>

### URL format

There are a few parameters that are required for all Shopping API requests. Here's simple request URL:

```
https://www.googleapis.com/shopping/search/v1/  
cx:016458501645884057912:dq_ixbwhuk8/products?  
key=<DEVKEY>&country=us&alt=atom&q=shirt
```

The following table lists the required fields in this URL.

<code>cx:016458501645884057912:dq_ixbwhuk8</code> (This one is for the Google Store's products)	Your Unique Search Engine Identifier This ties the query to your GCS control panel and Merchant Center Feed.
<code>key=&lt;DEVKEY&gt;</code>	This must contain your API key (from <a href="https://code.google.com/apis/console/">https://code.google.com/apis/console/</a> )
<code>country=us</code>	The API currently returns offers only from

	one country in one language and one currency per request.
--	---

There are two other parameters, `alt` and `q`, that are not required, but are typically passed in all API requests. The `alt` parameter specifies the result format either `alt=atom` or `alt=json` can be specified to return results in either XML (atom) or JSON format. The `q` parameter is used for full text search--terms in the `q` parameter are matched against all properties of all items in your feed.

Verify that the above URL works with your CX and API Key, and that you get results back from your product feed. Now we'll look at some of the other parameters used in API requests--try some of these parameters with your own data to familiarize yourself with how the parameters are specified and how they affect the returned results.

### Result size and pagination

The `maxResults` parameter controls how many results are returned by the request. This number can range from 0 to 1000. The default (if parameter is omitted from the request) is 25. Depending on your site, you'll probably want to customize this--anywhere from 20 to 100 results are common.

The option to return 1000 results should typically not be used on all search pages. Returning a large number of results is most often used on "View All" pages, when you want to allow the user to see all search results on a single page, rather than clicking through page after page of results. There is no way to return more than 1000 results.

Pagination is accomplished with the `startIndex` parameter and the `totalItems` and `itemsPerPage` elements returned in the result set. With these values you know how many items there are in the response, and you can page through them based on how many items per page you wish to display. The response also include the `nextLink` and `previousLink` elements which can be used for easy next/previous navigation.

### Sorting

By default, results are returned sorted by relevancy. There are a few ways you can affect sorting: If you want to still use Google's ranking algorithm, but bias some items higher or lower you can use the Ranking Rules functionality in the control panel. This is useful when you want to just bias the results, but not do a "hard-sort." Examples include promoting best sellers, popular brands, or demoting out-of-stock items.

Relevancy sorting is usually the best default sort option. Then you can also provide options for users to sort their results on other fields (price, rating, size, and so on). This is accomplished with the `rankBy` parameter.

### Content modules

The Shopping API has several "content modules," which enable different features in the result set. Currently there are five content modules available:

- Categories
- Facets
- Promotions

- Spelling
- Recommendations

Each module is enabled with the `module-name.enabled` parameter. Each also has another parameter, `module-name.useGcsConfig`, which when set to `true`, specifies that you want to use the configuration for that module as set in the GCS Control Panel. Some modules also have other parameters to control the results. For more details, see the content modules documentation at

<http://code.google.com/apis/shopping/search/v1/reference-content-modules.html>

Each content module, when enabled, returns its content in a separate block of data within the result set.

### Facets and refinements

The facets content module allows augmenting the response with facets. A *facet* is a histogram of the values of a specified property or attribute among all the products matching the search criteria. Facets usually appear in the left navigation bar, and allow users to refine results based on attributes of your items (color, size, price, and so on).

Enabling the facets module brings back a list of attributes, along with the values and counts for each of them. You can specify a list of facets to return using `facets.include`, or use `facets.discover` to return the most used facets for the given query automatically. In most cases using `facets.include` is the best choice because it gives you more control over the number of bucket values to return. But you can use both--specifying a list of facets to return always, and then use `facets.discover` to return also any other facets that might be relevant.

These parameters, coupled with the facet rules functionality in the control panel allow for great flexibility in determining which facets to display for a particular query.

### Refinements

Once you have the facet list returned, you can use them to provide options for refining the search results based on those facets.

A simple facet result for the color attribute might look like this (xml response):

```
<s:facet name="color" displayName="Color" type="text" count="5">
<s:bucket value="Red" count="2"/>
<s:bucket value="Green" count="1"/>
<s:bucket value="Blue" count="1"/>
</s:facet>
```

The results are easy enough to parse and display, but to make them a refinement requires creating a URL that takes the facet name, type and value into account. Refinements are done using the `restrictBy` parameter. So to refine by color and return only Red items in this case, we would construct a parameter like this: `&restrictBy=color(text)=Red`

You can also add more facets to the `restrictBy` parameter to further narrow your search. There are two ways to do this:

- [Specifying multiple values](#)
- [Specifying multiple facets](#)

### Specifying multiple values

You can allow users to choose multiple values with an OR operator. This will return items with *either* `color=Red` **OR** `color=Green`:

```
&restrictBy=color(text)=Red|Green
```

This is implemented when you allow users to check multiple values for a refinement.

### Specifying multiple facets

Refinements are additive, so if you specify multiple `restrictBy` fields they will be ANDed. This will return items with *both* `color=Red` **AND** `color=Green`:

```
&restrictBy=color(text)=Red,color(text)=Green
```

Typically this method is used when you allow users to select only one options per refinement. Here someone might have selected `size=small`, then `color=red` and `price` under \$20:

```
&restrictBy=size(text)=small,color(text)=red,price=[0,20]
```

### Specifying ranges

Range-based restrictions can be specified for numeric values. As shown in the example above, a price range restrict is specified as `&restrictBy=price=[0,20)`

The square brackets, `[ or ]`, mean “inclusive,” while the parentheses, `( or )`, mean “exclusive.”

`price=[0,20)` matches everything from 0 up to 20, while `price=(0,20]` matches everything greater than 0 up to and including 20.

### Breadcrumbs

“Breadcrumbs” are a user interface feature that keeps track of what options a user has selected, or levels of category navigation. Currently there are no features that explicitly provide breadcrumbs in the Shopping API, but they can still be implemented.

To generate the breadcrumb list your code would look at either the actual URL for your search page, or the “self” link in the results--both of which include all of the refinement parameters. Simply iterate through the parameter list and create a breadcrumb link for each parameter. When a user clicks on the breadcrumb, your code would remove that parameter and resubmit the API request.

In this query string example we can see the user has clicked on size and color refinements:

```
?q=android shirt&size=Large&color=Black
```

So a breadcrumb display might look something like:

[Home](#) / [android shirt](#) / [Large](#) / [Black](#)

This could be enhanced to have checkboxes next to each option to allow for removing that restriction.

Category breadcrumbs (parents of current category) can be returned by way of the category module. For more information, see:

<http://code.google.com/apis/commercesearch/docs/browse.html#breadcrumbs>

## Response format

The Shopping API returns results in XML (Atom) or JSON format. This is controlled with the `alt` parameter (`alt=atom` or `alt=json`). Most of our example code uses XML because XML makes for better human-readable examples, but the choice for your own implementation is up to you. JSON might provide a slight performance advantage because it is more compact and easier to parse.

Regardless of the response format, the content is the same. Each response contains some query-specific data (number of items returned, starting index, and so on), followed by a section for each of the content modules, followed by the list of items returned.

For detailed information and example responses, see “Reference - Response Format” at

<http://code.google.com/apis/shopping/search/v1/reference-response-format.html>

## Specifying fields to return

The default response contains a lot of data--probably more than is needed for simply presenting a search results page. For example, each item contains these attributes in the results:

```
<s:googleId>1575044745368379202</s:googleId>
  <s:providedId>123</s:providedId>
  <s:author>
    <s:name>Example Company</s:name>
    <s:accountId>12345</s:accountId>
  </s:author>
  <s:creationTime>2008-04-02T14:16:11.000Z</s:creationTime>
  <s:modificationTime>2011-06-29T07:19:09.000Z</s:modificationTime>
```

This adds a lot of extra data to the result set, which is usually not needed for displaying search results.

**Best Practice:** Limit the response to include only the fields and attributes you need. This improves performance by reducing the amount of data Google needs to process, as well as making the response payload smaller and easier to parse.

There are three parameters which control which are used to specify the data you wish to return: `fields`, `productFields` and `attributeFilter`.

Use the `fields` parameter to specify all the particular fields you need. The naming convention used for the `fields` parameter is slightly different for JSON vs. XML.

JSON:

```
&fields=spelling,promotions,redirects,selfLink,totalItems,startIndex,itemsPerPage,currentItemCount,categories, facets, items (product/title,product/attributes,product/link,product/images,product/inventories,product/categories)
```

**XML:**

```
&fields=openSearch:*,s:facets,entry(title,s:product/  
s:attributes,s:product/s:providedId,s:product/s:link)
```

Use the `attributeFilter` parameter to specify which of an item's attributes you need:

```
&attributeFilter=name(text),thumbnail(text),product+review+average(float),  
condition(text),price(float),brand(text)
```

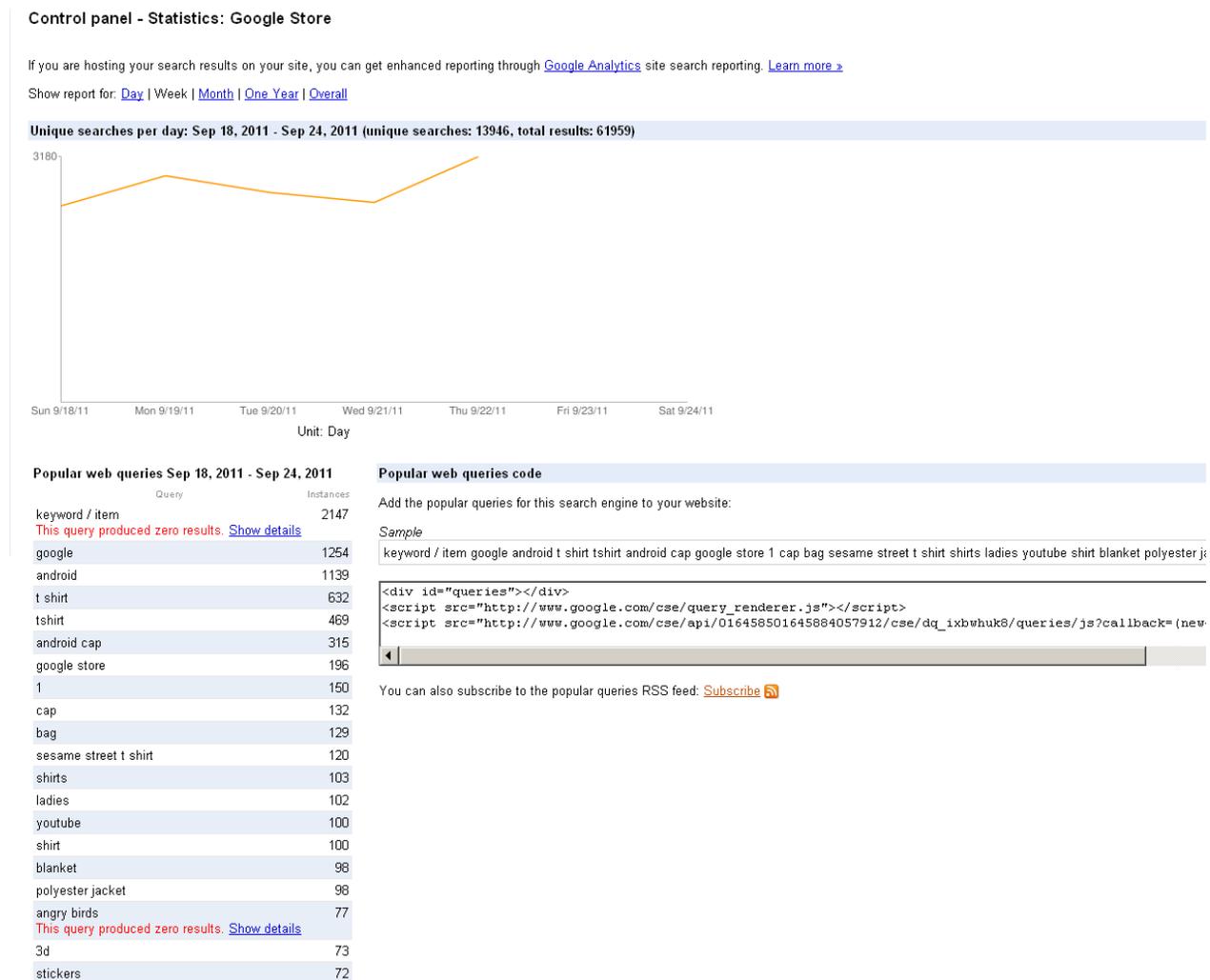
**Best Practice:** Limit the number of results returned for your default searches. You can provide the option for users to retrieve 50, 100 or even 1000 results--but these requests can take longer to return. Google is optimized to return small result sets very quickly--stick to returning 10 to 30 results per page by default.

## Chapter 5 GCS Stats and Analytics

Google Analytics (GA) is the preferred method for analyzing GCS performance on your site, although you can use any third-party analytics package. Google Analytics does require some additional configuration, but before we look at implementing GA there are a few ways to get some basic GCS statistics without any extra code.

### GCS Statistics page

The GCS Statistics page within the control panel provides very basic statistics. You can select fixed date ranges of Day (previous day)/Week (current week)/Month (current month)/Past Year/Since Creation, and view the top queries for those ranges.



The count for unique searches is the total number of unique queries, and the total number of results is the total number of results returned by all queries. The total results number is approximately the average number of results per page x the total number of searches.

Also shown are popular query terms.

These numbers do take into account SAYT and Autocompletion queries, which are not counted towards yearly GCS quota.

## API Console Traffic Reports

Another place to find some usage statistics is in the Traffic Reports section of the API console: <https://code.google.com/apis/console/>

To view the Traffic Reports, click the Search API for Shopping link (should show a status of “ON”).

This graph simply shows the total number of API requests for the given time period. As mentioned in [Chapter 1](#), it is a good idea to have separate keys for your server-side component, and any client-side code (SAYT/Autocomplete). The Traffic Reports have separate reports for each key, so you can get a better view of where API requests are being used.

## Google Analytics integration

If you’re already using Google Analytics on your site, getting useful data from GCS queries is easy. If you’re not currently using GA, see the “Getting Started Guide” for Analytics at <http://www.google.com/support/analytics/>.

There is currently no direct integration between GCS and GA. You can implement Google Analytics by:

1. Installing the Google Analytics Tracking code on your pages.
2. Configuring Site Search settings in the Analytics control panel.

While you could install the tracking code only on search results pages, it is recommended to install the tracking code on all pages of your site. The Recommendations feature in GCS relies on Google Analytics data for providing recommendations, and to measure the true impact of GCS, it is important to track conversion results.

## Configuring Site Search

GCS usage on your site is tracked in the Site Search section of Google Analytics. Once you have the tracking code installed on your site, follow the instructions for configuring Site Search tracking at <https://www.google.com/support/googleanalytics/bin/topic.py?topic=12626>.

The most important thing to capture is the Query Parameter. This is what is passed to the `q` parameter in the Shopping API request, although your own form and server-side code could have it named anything (keyword, substring, and so on).

The Category parameter can be used to log refinements. Typically this would be for a category refinement, but it could be used for any attribute. The results show up in the **Site Search > Categories** report, giving a list of the top category refinements.

With just this data you’ll be able to get valuable reports, such as:

- Top query terms

- Percentage of visitors that used search
- Percentage of searches that were refined
- Number of pageviews per search
- Percentage of search exits

But wait, that's not all! Even more important is seeing how well GCS usage results in conversions. Typically this means a sale, but you could also set up other goals such as a lead generated, or contact request initiated. So, if you're not already tracking e-commerce conversions, you should be.

### Configuring conversion tracking

Also important is tracking conversions and order values. In the same **Profile Settings** panel there is a radio button for **E-Commerce Website**. Make sure that **Yes** is selected. This enables the Ecommerce reports in Analytics. To get meaningful information here, you need to be sure to use the Analytics tracking code on all of your shopping cart and checkout pages, especially the "order complete" page.

Tracking conversions also requires some extra GA code to log the order amounts and the items purchased. To get started with tracking conversions, see "Ecommerce Tracking" at <http://code.google.com/apis/analytics/docs/tracking/gaTrackingEcommerce.html>

Once GA is implemented, you'll be able to get much more useful information about how GCS is performing on your site. Some common metrics are:

- Top converting searches and keywords
- Conversion rate for site search
- Average order value for site search
- Percentage of orders resulting from site search

These reports are especially important when making changes to GCS--by making incremental changes, and reviewing Analytics reports, you can find which changes had the best impact on performance. For example, you could make a change to the search results UI, and compare the data from the previous week to see if the change helped. True A/B or "split" testing is somewhat more difficult--you can use URL tagging to compare different versions of the same page, or Google's Website Optimizer tool to perform the tests:

<http://www.google.com/support/googleanalytics/bin/answer.py?hl=en-GB&answer=55589>

### Zero results

The standard reports in Google Analytics provide lots of great information, but the standard tracking code can only do so much--it is only able to log information that is in the page URL, and actual content from the page doesn't normally get saved to GA. One situation where you might want to feed GA a little extra information is when GCS doesn't find any matching items (zero results returned).

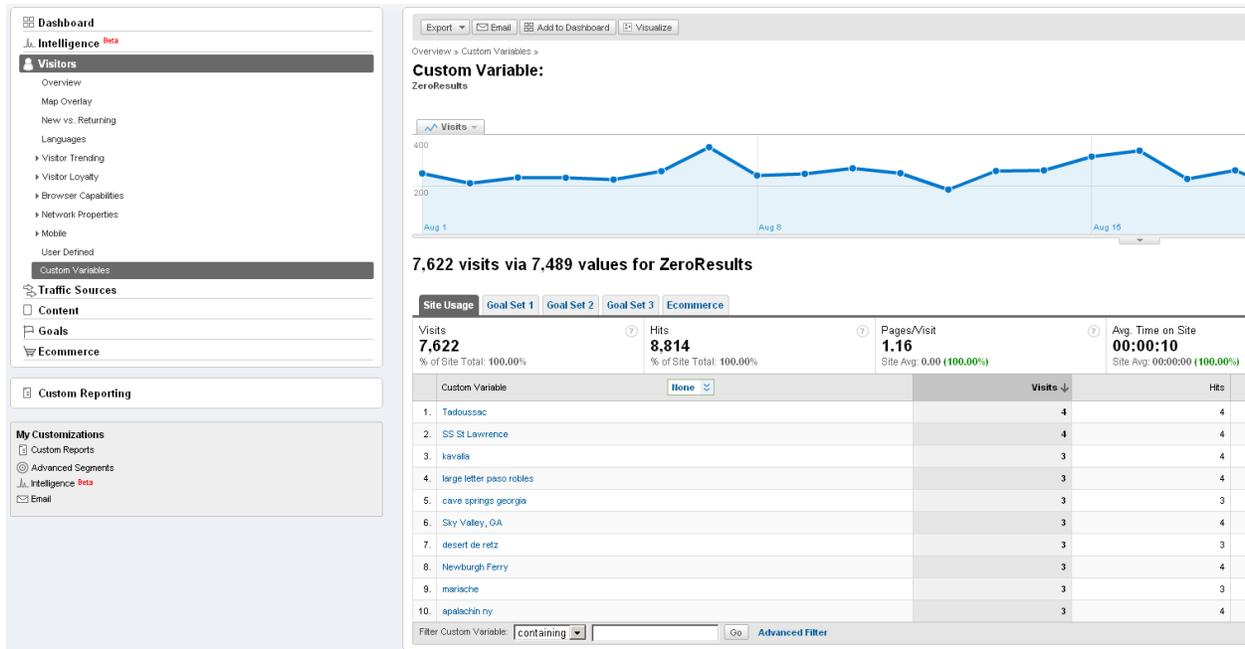
One way to log "extra" info to GA is by using Custom Variables:

<http://code.google.com/apis/analytics/docs/tracking/gaTrackingCustomVariables.html>

For logging Zero results, you would just add this line to the tracking code when GCS returns `totalResults = 0`:

```
_gaq.push(['_setCustomVar', 1, 'ZeroResults', '<query term here>', 3]);
```

This creates a custom variable named “ZeroResults” and you’ll get a report of all the query terms under **Visitors > Custom Variables**.



It’s not unusual to have quite a few zero results queries, but they should be a relatively small percentage of your total queries. You also want to be on the lookout for a large number of the same query terms. This could be a good opportunity for adding new items that you don’t currently have in your inventory. Looking at these reports is also a good way to find potential keywords for synonyms and promotions.

### Tracking other GCS features

The Site Search section in Google Analytics is meant for tracking queries entered by users, but there are some GCS features that are a bit different than normal site search, yet just as important for tracking conversions and other data.

Some features you might want to build reports for are:

- Refinements
- Navigation/Browse
- Promotions
- Recommendations

The easiest way to track these types of clicks is by adding a custom tracking parameter to the URL used in any of these features.

For promotions, a link is typically displayed to a product page or other landing page. To track this, you could rely on the referrer being set to the search results page, but you wouldn’t necessarily know if it were a click on a promotion or a item in the search results. So, add a

tracking parameters--something like `?gaclick=promo`--to the end of the URL. Now you have the ability to generate a report that shows how many times a promotion link was clicked.

Navigation/Browse typically doesn't need extra tracking parameters in the URL to report on Navigation usage (separate from search), unless you are using the same URL for site search and navigation.

Although the URL parameter method is easy, it's not always recommended. One problem is that search engines might index duplicate content, so you should exclude those pages with tracking codes with robots meta tags.

### Event tracking

A better, but slightly more complex way is to track the actual clicks, rather than the page view. This is accomplished using Google Analytics Event Tracking feature.

Event tracking allows you to log data to Google Analytics, independent of a pageview. This is useful for tracking things like refinements or clicks on recommendations.

Assuming you have the `ga.js` tracking code already on your search results page, you can add `onClick` events to any of the refinement links:

```
<a href="/search.php?substring=shirts&brand=Acme"
  onClick="_gaq.push(['_trackEvent', 'SearchRefine', 'Brand', 'Acme']
);">Acme</a>
```

This logs an event of type "SearchRefine" with action "Brand" and value "Acme". Alternatively, you can include a fourth value:

```
<a href="/search.php?substring=shirts&brand=Acme"
onClick="_gaq.push(['_trackEvent', 'SearchRefine', 'Brand', 'Acme', 'shirts']);">Acme</a>
```

This logs "shirts" as the query term associated with that refinement.

Adding these onclick events to all refinement URLs will allow a report to be generated showing the top types of refinements, as well as showing which refinements converted the most.

For product recommendation tracking, a similar approach can be used:

```
onClick="_gaq.push(['_trackEvent', 'Recommendation', 'Click', '146419', '3']);"
```

This would log a click on a recommended product, id # 146419, in position 3.

For more details on Event Tracking view the "Event Tracking Guide" at <http://code.google.com/apis/analytics/docs/tracking/eventTrackerGuide.html>

## Appendix A Limits

### Items in general

Items automatically get deleted in 30 days. As a result, in order to remain in the index, each item needs to be refreshed once every 30 days.

Item update can take up to 24 hours to take effects

### Feed size

- 10 feeds per account
- 100K items per feed (can be raised by Google if necessary)
- 10MB upload limit through Merchant Center or HTTPFetch
- 1GB Limit for FTP file

### Content API limit

- Content API has a 1MB limit per upload (b/4274808)

### Search limit

- No synonym support for attributes (b/4396415)
- No support for authentication (people who know your CX number can access your index)

## Appendix B GCS Launch Checklist

Customer Name:

Production CX:

Merchant Center ID:

API Key:

SAYT API Key:

<b>Merchant ID in Whitelist</b>	
<b>Visible Usecases &amp; Category Feed Enabled in ICS</b>	
<b>GCS Account Properly Provisioned</b>	
<b>Quota set for API Key(s)</b>	
<b>API Per-User-Limit (set in API Console "Traffic Controls")</b>	
<b>Whitelist for any Labs features</b>	
<b>Support Portal account active</b>	